Basic Feature Engineering With Time Series Data in Python

by **Jason Brownlee** on December 14, 2016 in **Time Series**

Last Updated on September 15, 2019

Time Series data must be re-framed as a supervised learning dataset before we can start using machine learning algorithms.

There is no concept of input and output features in time series. Instead, we must choose the variable to be predicted and use feature engineering to construct all of the inputs that will be used to make predictions for future time steps.

In this tutorial, you will discover how to perform feature engineering on time series data with Python to model your time series problem with machine learning algorithms.

After completing this tutorial, you will know:

- The rationale and goals of feature engineering time series data.
- How to develop basic date-time based input features.
- How to develop more sophisticated lag and sliding window summary statistics features.

Discover how to prepare and visualize time series data and develop autoregressive forecasting models in my new book, with 28 step-by-step tutorials, and full python code.

Let's dive in.

- **Updated Jun/2017**: Fixed a typo in the expanding window code example.
- **Updated Apr/2019**: Updated the link to dataset.
- **Updated Aug/2019**: Updated data loading to use new API.
- **Updated Sep/2019**: Fixed bug in data loading.

Basic Feature Engineering Wi
Photo by José Morcillo Valen

Feature Engineering for Time Series

A time series dataset must be transformed to be modeled as a supervised learning problem.

That is something that looks like:

```
time 1, value 1
time 2, value 2
time 3, value 3
```

To something that looks like:

```
input 1, output 1
input 2, output 2
input 3, output 3
```

So that we can train a supervised learning algorithm.

Input variables are also called features in the field of machine learning, and the task before us is to create or invent new input features from our time series dataset. Ideally, we only want input features that best help the learning methods model the relationship between the inputs (**X**) and the outputs (**y**) that we would like to predict.

In this tutorial, we will look at three classes of features that we can create from our time series dataset:

1. **Date Time Features**: these are components of the time step itself for each observation.
2. **Lag Features**: these are values at prior time steps.
3. **Window Features**: these are a summary of values over a fixed window of prior time steps.

Before we dive into methods for creating input features from our time series data, let's first review the goal of feature engineering.

Goal of Feature Engineering

The goal of feature engineering is to provide strong ar
features and the output feature for the supervised lear

In effect, we are are moving complexity.

Complexity exists in the relationships between the input and output data. In the case of time series, there is no concept of input and output variables; we must invent these too and frame the supervised learning problem from scratch.

We may lean on the capability of sophisticated models to decipher the complexity of the problem. We can make the job for these models easier (and even use simpler models) if we can better expose the inherent relationship between inputs and outputs in the data.

The difficulty is that we do not know the underlying inherent functional relationship between inputs and outputs that we're trying to expose. If we did know, we probably would not need machine learning.

Instead, the only feedback we have is the performance of models developed on the supervised learning datasets or "views" of the problem we create. In effect, the best default strategy is to use all the knowledge available to create many good datasets from your time series dataset and use model performance (and other project requirements) to help determine what good features and good views of your problem happen to be.

For clarity, we will focus on a univariate (one variable) time series dataset in the examples, but these methods are just as applicable to multivariate time series problems. Next, let's take a look at the dataset we will use in this tutorial.

Minimum Daily Temperatures Dataset

In this post, we will use the Minimum Daily Temperatures dataset.

This dataset describes the minimum daily temperatures over 10 years (1981-1990) in Melbourne, Australia.

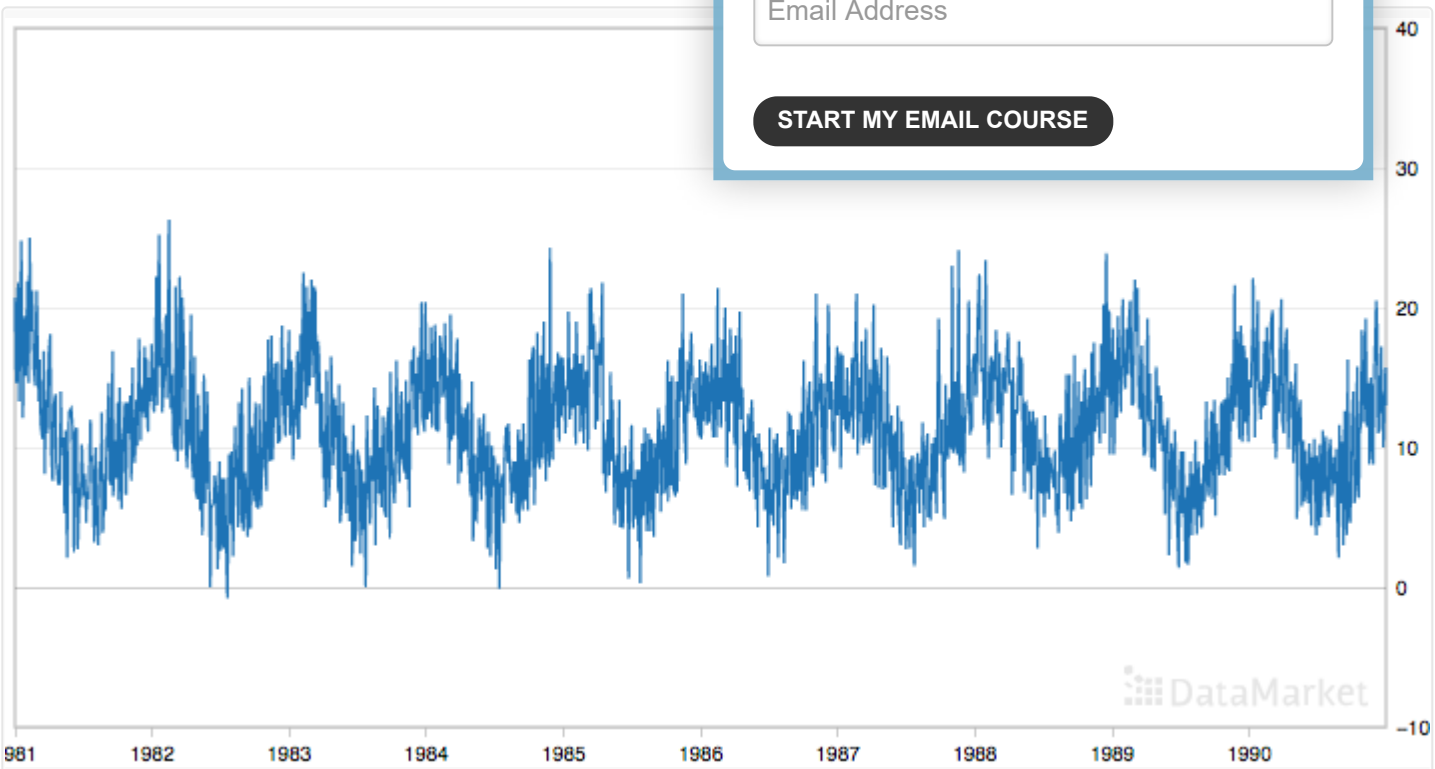
The units are in degrees Celsius and there are 3,650 observations. The source of the data is credited as the Australian Bureau of Meteorology.

- Download the dataset.

Below is a sample of the first 5 rows of data, including the header row.

```
"Date","Temperature"
"1981-01-01",20.7
"1981-01-02",17.9
"1981-01-03",18.8
"1981-01-04",14.6
"1981-01-05",15.8
```


Below is a plot of the entire dataset.



Minimum Daily Temperatures

The dataset shows an increasing trend and possibly some seasonal components.

- Download the dataset.

Date Time Features

Let's start with some of the simplest features that we can use.

These are features from the date/time of each observation. In fact, these can start off simply and head off into quite complex domain-specific areas.

Two features that we can start with are the integer month and day for each observation. We can imagine that supervised learning algorithms may be able to use these inputs to help tease out time-of-year or time-of-month type seasonality information.

The supervised learning problem we are proposing is to predict the daily minimum temperature given the month and day, as follows:

```
Month, Day, Temperature
Month, Day, Temperature
Month, Day, Temperature
```

We can do this using Pandas. First, the time series is l v
Pandas *DataFrame* for the transformed dataset.

Next, each column is added one at a time where mont
stamp information for each observation in the series.

Below is the Python code to do this.

```
# create date time features of a dataset
from pandas import read_csv
from pandas import DataFrame
series = read_csv('daily-minimum-temperatures.csv', header=0, index_col=0, parse_dates=True, squ
dataframe = DataFrame()
dataframe['month'] = [series.index[i].month for i in range(len(series))]
dataframe['day'] = [series.index[i].day for i in range(len(series))]
dataframe['temperature'] = [series[i] for i in range(len(series))]
print(dataframe.head(5))
```

Running this example prints the first 5 rows of the transformed dataset.

```
month day temperature
0 1 1 20.7
1 1 2 17.9
2 1 3 18.8
3 1 4 14.6
4 1 5 15.8
```

Using just the month and day information alone to predict temperature is not sophisticated and will likely result in a poor model. Nevertheless, this information coupled with additional engineered features may ultimately result in a better model.

You may enumerate all the properties of a time-stamp and consider what might be useful for your problem, such as:

- Minutes elapsed for the day.
- Hour of day.
- Business hours or not.

- Weekend or not.
- Season of the year.
- Business quarter of the year.
- Daylight savings or not.
- Public holiday or not.
- Leap year or not.

From these examples, you can see that you're not restricted to the raw integer values. You can use binary flag features as well, like whether or not the observation was recorded on a public holiday.

In the case of the minimum temperature dataset, maybe the season would be more relevant. It is creating domain-specific features like this that are more likely to add value to your model.

Date-time based features are a good start, but it is ofte ous time steps. These are called lagged values and we wi

Lag Features

Lag features are the classical way that time series fore d learning problems.

The simplest approach is to predict the value at the ne (t-1). The supervised learning problem with shifted value

```
Value(t-1), Value(t+1)
Value(t-1), Value(t+1)
Value(t-1), Value(t+1)
```

The Pandas library provides the shift() function to help create these shifted or lag features from a time series dataset. Shifting the dataset by 1 creates the t-1 column, adding a NaN (unknown) value for the first row. The time series dataset without a shift represents the t+1.

Let's make this concrete with an example. The first 3 values of the temperature dataset are 20.7, 17.9, and 18.8. The shifted and unshifted lists of temperatures for the first 3 observations are therefore:

```
Shifted, Original
NaN, 20.7
20.7, 17.9
17.9, 18.8
```

We can concatenate the shifted columns together into a new DataFrame using the concat() function along the column axis (*axis=1*).

Putting this all together, below is an example of creating a lag feature for our daily temperature dataset. The values are extracted from the loaded series and a shifted and unshifted list of these values is created. Each column is also named in the *DataFrame* for clarity.

```
from pandas import read_csv
from pandas import DataFrame
from pandas import concat
```

```
series = read_csv('daily-min-temperatures.csv', header=0, index_col=0)
temps = DataFrame(series.values)
dataframe = concat([temps.shift(1), temps], axis=1)
dataframe.columns = ['t-1', 't+1']
print(dataframe.head(5))
```

Running the example prints the first 5 rows of the new dataset with the lagged feature.

```
t-1 t+1
0 NaN 20.7
1 20.7 17.9
2 17.9 18.8
3 18.8 14.6
4 14.6 15.8
```

You can see that we would have to discard the first row to use the dataset to train a supervised learning model, as it does not contain enough data to work with

The addition of lag features is called the sliding windo is as though we are sliding our focus along the time serie at is within the window width.

We can expand the window width and include more la ase modified to include the last 3 observed values to predi


```
from pandas import read_csv
from pandas import DataFrame
from pandas import concat
series = read_csv('daily-min-temperatures.csv', header=0, index_col=0)
temps = DataFrame(series.values)
dataframe = concat([temps.shift(3), temps.shift(2), temps.shift(1), temps], axis=1)
dataframe.columns = ['t-3', 't-2', 't-1', 't+1']
print(dataframe.head(5))
```

Running this example prints the first 5 rows of the new lagged dataset.

```
t-3 t-2 t-1 t+1
0 NaN NaN NaN 20.7
1 NaN NaN 20.7 17.9
2 NaN 20.7 17.9 18.8
3 20.7 17.9 18.8 14.6
4 17.9 18.8 14.6 15.8
```

Again, you can see that we must discard the first few rows that do not have enough data to train a supervised model.

A difficulty with the sliding window approach is how large to make the window for your problem.

Perhaps a good starting point is to perform a sensitivity analysis and try a suite of different window widths to in turn create a suite of different “views” of your dataset and see which results in better performing models. There will be a point of diminishing returns.

Additionally, why stop with a linear window? Perhaps you need a lag value from last week, last month, and last year. Again, this comes down to the specific domain.

In the case of the temperature dataset, a lag value from the same day in the previous year or previous few years may be useful.

We can do more with a window than include the raw values. In the next section, we'll look at including features that summarize statistics across the window.

Rolling Window Statistics

A step beyond adding raw lagged values is to add a summary of the values at previous time steps.

We can calculate summary statistics across the values in the sliding window and include these as features in our dataset. Perhaps the most useful is the mean of the previous few values, also called the rolling mean.

For example, we can calculate the mean of the previo ue. For the temperature data, we would have to wait 3 tim ge of before we could use that value to predict a 3rd valu

For example:

```
mean(t-2, t-1), t+1
mean(20.7, 17.9), 18.8
19.3, 18.8
```

Pandas provides a rolling() function that creates a nev time step. We can then perform statistical functions on the window of values collected for each time step, such as calculating the mean.

First, the series must be shifted. Then the rolling dataset can be created and the mean values calculated on each window of two values.

Here are the values in the first three rolling windows:

```
#, Window Values
1, NaN
2, NaN, 20.7
3, 20.7, 17.9
```

This suggests that we will not have usable data until the 3rd row.

Finally, as in the previous section, we can use the *concat()* function to construct a new dataset with just our new columns.

The example below demonstrates how to do this with Pandas with a window size of 2.

```
from pandas import read_csv
from pandas import DataFrame
from pandas import concat
series = read_csv('daily-min-temperatures.csv', header=0, index_col=0)
temps = DataFrame(series.values)
shifted = temps.shift(1)
```

```
window = shifted.rolling(window=2)
means = window.mean()
dataframe = concat([means, temps], axis=1)
dataframe.columns = ['mean(t-2,t-1)', 't+1']
print(dataframe.head(5))
```

Running the example prints the first 5 rows of the new dataset. We can see that the first two rows are not useful.

- The first NaN was created by the shift of the series.
- The second because NaN cannot be used to calculate a mean value.
- Finally, the third row shows the expected value of 19.30 (the mean of 20.7 and 17.9) used to predict the 3rd value in the series of 18.8.

```
mean(t-2,t-1) t+1
0 NaN 20.7
1 NaN 17.9
2 19.30 18.8
3 18.35 14.6
4 16.70 15.8
```

There are more statistics we can calculate and even d
definition of the “window.”

Below is another example that shows a window width
statistics, specifically the minimum, mean, and maxim

You can see in the code that we are explicitly specifyi
lets us use it both in calculating the correct shift of the series and in specifying the width of the window to the *rolling()* function.

In this case, the window width of 3 means we must shift the series forward by 2 time steps. This makes the first two rows NaN. Next, we need to calculate the window statistics with 3 values per window. It takes 3 rows before we even have enough data from the series in the window to start calculating statistics. The values in the first 5 windows are as follows:

```
#, Window Values
1, NaN
2, NaN, NaN
3, NaN, NaN, 20.7
4, NaN, 20.7, 17.9
5, 20.7, 17.9, 18.8
```

This suggests that we would not expect usable data until at least the 5th row (array index 4)

```
from pandas import read_csv
from pandas import DataFrame
from pandas import concat
series = read_csv('daily-min-temperatures.csv', header=0, index_col=0)
temps = DataFrame(series.values)
width = 3
shifted = temps.shift(width - 1)
window = shifted.rolling(window=width)
dataframe = concat([window.min(), window.mean(), window.max(), temps], axis=1)
dataframe.columns = ['min', 'mean', 'max', 't+1']
```

```
print(dataframe.head(5))
```

Running the code prints the first 5 rows of the new dataset.

We can spot check the correctness of the values on the 5th row (array index 4). We can see that indeed 17.9 is the minimum and 20.7 is the maximum of values in the window of [20.7, 17.9, 18.8].

```
min mean max t+1
0 NaN NaN NaN 20.7
1 NaN NaN NaN 17.9
2 NaN NaN NaN 18.8
3 NaN NaN NaN 14.6
4 17.9 19.133333 20.7 15.8
```

Expanding Window Statistics

Another type of window that may be useful includes al

This is called an expanding window and can help with ...like the *rolling()* function on *DataFrame*, Pandas provides ...or values for each time step.

These lists of prior numbers can be summarized and i... the lists of numbers in the expanding window for the first 5


```
#, Window Values
1, 20.7
2, 20.7, 17.9,
3, 20.7, 17.9, 18.8
4, 20.7, 17.9, 18.8, 14.6
5, 20.7, 17.9, 18.8, 14.6, 15.8
```

Again, you can see that we must shift the series one-time step to ensure that the output value we wish to predict is excluded from these window values. Therefore the input windows look as follows:

```
#, Window Values
1, NaN
2, NaN, 20.7
3, NaN, 20.7, 17.9,
4, NaN, 20.7, 17.9, 18.8
5, NaN, 20.7, 17.9, 18.8, 14.6
```

Thankfully, the statistical calculations exclude the NaN values in the expanding window, meaning no further modification is required.

Below is an example of calculating the minimum, mean, and maximum values of the expanding window on the daily temperature dataset.

```
# create expanding window features
from pandas import read_csv
from pandas import DataFrame
from pandas import concat
series = read_csv('daily-min-temperatures.csv', header=0, index_col=0)
temps = DataFrame(series.values)
window = temps.expanding()
```

```
dataframe = concat([window.min(), window.mean(), window.max(), temps.shift(-1)], axis=1)
dataframe.columns = ['min', 'mean', 'max', 't+1']
print(dataframe.head(5))
```

Running the example prints the first 5 rows of the dataset.

Spot checking the expanding minimum, mean, and maximum values shows the example having the intended effect.

```
    min       mean   max   t+1
0  20.7  20.700000  20.7  17.9
1  17.9  19.300000  20.7  18.8
2  17.9  19.133333  20.7  14.6
3  14.6  18.000000  20.7  15.8
4  14.6  17.560000  20.7  15.8
```


Summary

In this tutorial, you discovered how to use feature engi
supervised learning dataset for machine learning.

Specifically, you learned:

- The importance and goals of feature engineering
- How to develop date-time and lag-based features
- How to develop sliding and expanding window su

Do you know of more feature engineering methods for time series?
Let me know in the comments below.

Do you have any questions?
Ask your questions in the comments below and I will do my best to answer.

About Jason Brownlee

Jason Brownlee, PhD is a machine learning
modern machine learning methods via han
View all posts by Jason Brownlee →

‹ How to Normalize and Standardize Time Series Data in Python

How To Resample and Interpolate Your Time Series Data With Python ›

82 Responses to *Basic Feature Engineering With Time Series Data in Python*

Dario Romero December 14, 2016 at 9:11 am #

REPLY

Awesome introduction to Feature Engineering with Time Series. I thought that it was something only for scientific minds but actually it is so easy. I enjoyed the easy way that you have explained it. I am ready now for any challenge using Time Series in my domain of expertise. Thanks!!!!

Jason Brownlee December 15, 2016 at 8:27 am #

REPLY

I'm glad to hear it Dario.

fai December 15, 2016 at 2:31 pm #

REPLY

min mean max t+1
0 NaN NaN NaN 20.7
1 NaN NaN NaN 17.9
2 NaN NaN NaN 18.8
3 NaN NaN NaN 14.6
4 17.9 19.133333 20.7 15.8

why you shift 2 step instead of 1? to make it

min mean max t+1
0 NaN NaN NaN 20.7
1 NaN NaN NaN 17.9
2 NaN NaN NaN 18.8
3 17.9 19.133333 20.7 14.6

great post btw… thanks !!

Jason Brownlee December 16, 2016 at 5:37 a

Hi Fai, the shifting can get confusing.

I try to explain it above by showing how the groups of observations in the window look each row as we build them up. I guess it didn't work as an explanatory tool:

```
1, NaN
2, NaN, NaN
3, NaN, NaN, 20.7
4, NaN, 20.7, 17.9
5, 20.7, 17.9, 18.8
```

fai December 16, 2016 at 6:42 pm #

REPLY

Thx for the reply. Thats a good explanatory tool. But i was wondering

if shift by 1 step, we could have the following window observations,
and we can have enough data in row 4 for calculating statistic with 3 values per window.

1, NaN
2, NaN, 20.7
3, NaN, 20.7, 17.9
4, 20.7, 17.9, 18.8

so why should we shift by 2 step instead (we can only start calculating at row5)

1, NaN
2, NaN, NaN
3, NaN, NaN, 20.7
4, NaN, 20.7, 17.9
5, 20.7, 17.9, 18.8

Jason Brownlee December 17, 2016 at 11:11 am #

REPLY

If you're asking why a window size of "3" or a window size of "2" or even "1", no reason. Just for demonstration purposes.

We can use autocorrelation to find good window si
correlograms) and I will get into this future posts.

Does this help at all? Or am I still misunderstandin

Abdelrahman Hammad November 7,

Hi Jason,

Actually, I have the same concern. There are t

1. Rolling Window. The size is 3, and that's not the concern.
2. Lagging. The shift is 2, and the question is: why don't we use time steps 1, 2, and 3 to create features for time step 4? Why did we use time steps 1, 2, and 3 to create features in time step 5? By features I mean (min, max, & mean of 3 leading measurements)

Instead of: shifted = temps.shift(width – 1)
why don't we use: shifted = temps.shift(1)

Raghda Ziada August 21, 2019 at 2:36 am #

REPLY

sorry can u explain more how to use auto correlation for finding good window size?

Jason Brownlee August 21, 2019 at 6:49 am #

REPLY

Yes, see this post:
https://machinelearningmastery.com/gentle-introduction-autocorrelation-partial-autocorrelation/

Hicham Zmarrou January 4, 2017 at 11:50 pm #

REPLY

Hi Jason, nice introduction. If you are interested in more statistical features tale a look at http://rsif.royalsocietypublishing.org/content/10/83/20130048.full and the associted github repo.
I find it the most complete work for ts features engineering

REPLY

Jason Brownlee January 5, 2017 at 9:19 am #

Thanks for sharing Hicham.

REPLY

Paddy January 9, 2017 at 11:22 pm #

Hi,

In rolling mean why do you take lag of raw variables.wi

Jason Brownlee January 10, 2017 at 8:57 am

In this case, the data is univariate, there a
"invented" like lag vars.

REPLY

Daniel Kornhauser March 18, 2017 at 8:20 am #

Hi Jason:

When you write "predict the daily maximum temperature" did you mean "predict the daily minimum temperature" ?

REPLY

Jason Brownlee March 19, 2017 at 6:06 am #

I did Daniel, thanks. Fixed.

REPLY

Danny Ocean March 29, 2017 at 5:59 am #

I got error when reading the data. Where exactly did you spot the questionmarks?

Jason Brownlee March 29, 2017 at 9:12 am #

You can use a text editor and the find-replace feature.

nick May 14, 2017 at 12:38 am #

REPLY

how to create lagged variable for test data? we do not know any info in test data, right?

Jason Brownlee May 14, 2017 at 7:29 am #

REPLY

The lag for the test data will be the end of the train data.

You can prepare the data using the functio
http://machinelearningmastery.com/conver

Shaun November 22, 2019 at 2:30

Hi Jason,
Thank for great article.
May I ask which comes first?
Splitting Train/Test Sets on original Tin
Or
Splitting Train/Test Sets on Transformed (E.g. after being features engineered) dataset?

Please advise,
Shaun

Jason Brownlee November 22, 2019 at 6:10 am #

Great question.

Scaling is applied to each variable.
Transform to supervised happens before split.

But, you need to ensure that any scaling is learned on the training set only and apply to the test set – so there is no data leakage.

Not sure if that helps?

Farshid July 17, 2017 at 11:12 pm #

REPLY

It would be greate if you added some comments about forecasting time series with machine learning techniques like Nearest Neighbor or SVR methods.

REPLY

Jason Brownlee July 18, 2017 at 8:45 am #

Great suggestion, I hope to cover it in the future.

REPLY

vasil July 24, 2017 at 10:07 pm #

Hello !

There is a task for the Time Series Forecasting with the

Multiple input variables:
1 year
2. Month
3. Day of the week
4. Date of the month
5. Hour of the day
…
Hour of day.
Business hours or not.
Weekend or not.

Imprint – the number of sales

It is required to help – how to implement the algorithm in Python with Keras.
Thank you !

REPLY

Jason Brownlee July 25, 2017 at 9:42 am #

I would recommend using any of my many posts on LSTMs for time series as a starting point and adapt it to your problem.

REPLY

Ming February 23, 2018 at 9:34 am #

I love this sentence: complexity exists in the relationships between input and output data!

Because the output data may or may not be a straightforward outcome of the input data, what we are trying to do is to establish a predictable relationship between A and B. By feature engineering, we are hoping to reduce the complexity and difficulty of the establishment with intermediate features that may be more informative and closer to B.

We always hope to build a closer bridge to save time and energy.

Thank you so much for that punctual summary and great explanation! It is very inspiring and helpful.

By the way, what do you mean here “In the case of time series, there is no concept of input and output variables”? Aren’t we always provided a time series that comes with its time step (input) and its value (Y)? Do you mean concept of input and output variables for unknown new features we are going to invent?

Thanks.

Jason Brownlee February 23, 2018 at 12:05 pm #

REPLY

Thanks!

I guess I meant that we have really only have lags

Andy April 6, 2018 at 12:50 pm #

Hi, Jason! This is a great piece!

I have a question on how to sample this kind of data se

My understanding is that since the features are genera
among rows and across the variables. So it may require different treatment than the normal data sets which we assume the rows are independent between one another.

Could you please help to advise on the sampling method applied on this kind of data set?

Thank you very much!

Jason Brownlee April 6, 2018 at 3:50 pm #

REPLY

Yes, you can use walk-forward validation to evaluate models on sequence prediction problems.

See this post:
https://machinelearningmastery.com/backtest-machine-learning-models-time-series-forecasting/

Krystian April 17, 2018 at 7:50 am #

REPLY

Hi, Jason.

What do you think about combining Date Time Features with Lag Features? Is this approach justified?

Thanks a lot.

Jason Brownlee April 17, 2018 at 2:47 pm #

REPLY

Perhaps.

I would recommend doing the experiment and using the results/data to decide whether it is a good idea for your specific forecasting problem.

Prash April 17, 2018 at 10:43 am #

REPLY

Hello,
I have a time series data(csv), which is multimodal(obt
multiclass classification, by extracting features from thi
classification

Jason Brownlee April 17, 2018 at 2:49 pm #

Good question.

I don't have post on time series classification. I hop

Mark Roy May 28, 2018 at 2:00 pm #

REPLY

I'm having a bit of trouble with your terminology. You use "t-1" and "t+1" and refer to the use of the lagged "t-1" value to predict the "t+1" value using supervised machine learning. Wouldn't it be more accurate to say we use "t-1" to predict the value at "t"? And in using an ML algorithm, would the value at "t" be the "target variable"?

To clarify the context of my question, here is some background. I'm experimenting with an ML example to predict sales of items for a future month given history of sales. My training data has lagged monthly sales for the prior 3 months and for each training row, the target variable is the monthly sales for the current month. As I read your excellent post, I began wondering if I'm supposed to be adding one more feature to each training row… the sales for the NEXT month (t+1).

One follow up question is whether or not you have found it useful to add a "recent growth" feature. This in my example would highlight a hot new product or one that is beginning to fall off a cliff.

Thanks so much for a great article!

Jason Brownlee May 28, 2018 at 2:35 pm #

REPLY

Yes, t+1 should have been t. I cleaned up thi on the topic.

schwab July 5, 2018 at 6:18 am #

REPLY

you nailed it right. awesome material. Thanks.

Jason Brownlee July 5, 2018 at 8:04 am #

REPLY

Thanks.

Desi Mazdur August 1, 2018 at 5:56 pm #

Thank you for an excellent series of articles. I
and I always find that any questions that I have after g
answered in your blog posts.

There is one question which I have not found an answ
deviation or any other statistical derivative of the univa
data itself? Will this depend on the domain of the data
of stock prices as compared to the data of temperature
of the domain.

In other words, if I have to answer the question should I provide the mean of the data instead of the raw data (and assuming that the data is stock prices), should I look into research literature of machine learning or the literature of financial markets?

Thank you for any inputs

Jason Brownlee August 2, 2018 at 5:58 am #

REPLY

Yes, it depends on the data and on the method you are using to address it.

E.g. it may make sense for ML algorithms, but not for an ARIMA or ETS method.

As for stock prices, they are not predictable:
https://machinelearningmastery.com/faq/single-faq/can-you-help-me-with-machine-learning-for-finance-or-the-stock-market

Brian Stephans August 7, 2018 at 2:18 am #

REPLY

In the date time features I am getting the complete date 1981-01-01 in the temperature column any idea why. (note I am reading the file in as a read_csv a

Thanks
Brian

Jason Brownlee August 7, 2018 at 6:29 am #

REPLY

Perhaps this is a change in a recent version of Pandas?

Tina September 21, 2018 at 5:16 pm #

REPLY

If data is non-stationary, do we make transfor
variance and trend first before doing features?

Jason Brownlee September 22, 2018 at 6:25

You can use a power transform to adjust
trend/seasonality.

Eran October 23, 2018 at 12:41 am #

REPLY

Thank you for this great read.
I want each row in an hourly format, but the algorithm to predict what will be the value in 24 hours.
The reason for that is that hourly changes in the predicted variable are too small, but I do want to make a prediction even hour of the day.

1. Does it make sense to calculate a new column with `.shift(-24)` and use it as the predicted value?

2. Assuming you don't have enough temperatures data from Melbourne.
Assuming you want to use other cities to make a general model (that way you have more data).
Assuming the temperatures values are different, but they behave similar.
Does it make sense to use `.pct_change` instead of raw values (even for `max`, `min` and `mean`) calculations?

Jason Brownlee October 23, 2018 at 6:28 am #

REPLY

I'm not sure I follow, sorry.

Stefan November 12, 2018 at 9:26 pm #

REPLY

Thank you for a great article!

I was wondering when doing the shift, is it necessary to include the features for the “shifted” period too?

For example, say it’s a time series that is highly dependent on sales last year and whether it’s a Holiday or not, and I want to predict the sales.

Then I have one feature that is the sales for the same date last year and one feature that is Holiday (1/0).

If I want to predict the sales for 16th April 2018, it’s not a holiday so that feature would be 0 and I would just use the sales for the same date 2017. But the feature for sales last year, would be affected because the Easter was that date 2017.

Would I then need to add a feature like “same date last year Holiday” that is 1/0, or is that covered by the other features?

Hope that is understandable…
Thanks for an amazing blog!

Jason Brownlee November 13, 2018 at 5:45 a

Sure.

Parul Verma December 23, 2018 at 1:08 am #

Hey Jason,

while running the code for rolling window, it is giving the error:
“cannot handle this type -> object”

I am running the code for the dataset you have taken

Jason Brownlee December 23, 2018 at 6:06 am #

REPLY

Are you using Python 3 and statmodels 8+?

Magnus February 14, 2019 at 8:07 pm #

REPLY

Hi,

Thanks again for a good post. Do you know about Python Featuretools and Autokeras? It would be nice to see a comparison with these tools.

Jason Brownlee February 15, 2019 at 8:00 am #

REPLY

Thanks for suggestions Magnus.

Jon Nickerson March 13, 2019 at 5:19 am #

REPLY

Wow. Once again, you've explained the concepts perfectly.

After 5 other articles on this same topic, I now understand.

Very appreciative!

Jason Brownlee March 13, 2019 at 8:02 am #

I'm glad it helped Jon!

Shrouq Alelaumi April 19, 2019 at 2:20 am #

Hi Jason,
Thank you very much for this useful post. but please can you show how we can code non linear window as you mentioned here "Additionally, why stop with a linear window? Perhaps you need a lag value from last week, last month, and last year. Again, this comes down to the specific domain.
In the case of the temperature dataset, a lag value from the same day in the previous year or previous few years may be useful."

Thank you again!

Jason Brownlee April 19, 2019 at 6:18 am #

REPLY

Thanks for the suggestion.

Ryan Devera June 13, 2019 at 12:22 pm #

REPLY

Hey Jason!

Your tutorials are always very helpful! I had a quick question. If I just wanted to use a standard linear regression with these types of features. What are some common methods for filling in the NaN values on the shifted variables?

Jason Brownlee June 13, 2019 at 2:35 pm #

REPLY

The rows with NaNs must be dropped.

Ryan September 18, 2019 at 3:24 am #

REPLY

Okay, great thanks for the info. Do you have a tutorial that uses this concept and shows how to build the predictions? With a linear regressor or random forest for example? If not could you give a high level explanation of the steps that you might use? In the meantime I might write some code to get some feedback.

Jason Brownlee September 18, 20

I may, perhaps check here:
https://machinelearningmastery.com/start-

Perhaps this:
https://machinelearningmastery.com/multi-
models-for-household-electricity-consumpt

These are the steps for any project:
https://machinelearningmastery.com/start-here/#process

Valen June 21, 2019 at 3:31 pm #

REPLY

Hi Jason!
Thank you for the great article! It is really helpful. I have a question if you don't mind. Is there any method for determining the window size if I were to compute rolling window statistic? Thank you in advance!

Jason Brownlee June 22, 2019 at 6:30 am #

REPLY

Perhaps test a suite of different configurations to see what works well/best for your specific dataset and choice of model.

Jimmy Chiang August 5, 2019 at 3:26 am #

REPLY

Hi Jason!
First,thanks for the article. But I have some qu

series,that is t+1. But what if I need to predict the next two series or maybe more?

Jason Brownlee August 5, 2019 at 6:54 am #

REPLY

For a linear model like ARIMA, you can call forecast() or predict() and specify the number of step to forecast.

SwePalm August 16, 2019 at 6:50 pm #

REPLY

Noticed some questions that can be related b
series = Series.from_csv('daily-min-temperatures.csv',
FutureWarning: from_csv is deprecated. Please use re
I tested a few options to use read_csv, and this looks li
series = pd.read_csv('daily-min-temperatures.csv', inde

SwePalm August 16, 2019 at 6:53 pm #

Copy/paste error, should be:
series = pd.read_csv('daily-min-temperatures.csv',

Jason Brownlee August 17, 2019 at 5:34 am #

REPLY

Thanks for sharing.

teimoor September 15, 2019 at 5:34 am #

REPLY

hi i think this part of the code is wrong:

```
from pandas import read_csv
from pandas import DataFrame
series = read_csv('daily-min-temperatures.csv', header=0, index_col=0)
dataframe = DataFrame()
dataframe['month'] = [series.index[i].month for i in range(len(series))]
dataframe['day'] = [series.index[i].day for i in range(len(series))]
dataframe['temperature'] = [series[i] for i in range(len(series))]
print(dataframe.head(5))
```

it gives me the following error:
AttributeError: 'str' object has no attribute 'month'

Jason Brownlee September 15, 2019 at 6:31 am #

REPLY

Thanks, I have updated the example.

Colin October 29, 2019 at 4:18 am #

REPLY

Hi Jason,

First, thanks for the tutorial!

How do we decide the width of the window? Should we

Jason Brownlee October 29, 2019 at 5:31 am

Perhaps test a range of configurations an

Also an ACF/PACF plot can be insightful.

Mizy November 15, 2019 at 11:15 pm #

REPLY

These are helpful to me. Thanks a lot! Can I practice this and put it on my github? Obviously, the source link will upload together.

Jason Brownlee November 16, 2019 at 7:24 am #

REPLY

If it is not public.

David Guo January 7, 2020 at 3:56 pm #

REPLY

Hi Jason,

Thank you for this great work. I have a question and hope you don't mind. I am trying to building a weekly sales forecast model with Xgboost. My goal is using the model to predict the sales of future 8 weeks. But I am confused with the lag and diff features. I created many features like lag 2, lag 3, lag 4, lag 5 and diff 2, diff 3, diff 4,diff 5. It shows a great performance in my train set and test set based on historical data. Meanwhile I also checked the feature importance. Those lag and diff features rank top among all features. But when I want to predict the sales of future weeks. I don't have the actual sales, so many feature gonna to

be NaN and the predict result seems to be unreasonable. May I ask did you face this issue before? How should I deal with lag, diff feature on future time window? Thanks!

David

REPLY

Jason Brownlee January 8, 2020 at 8:19 am #

Good question.

Yes, you must chose your input variables (frame the prediction problem) based on the data that you do have available at the time you need to make a prediction. E.g. if you won't have specific lab obs at the time of prediction, then don't frame the problem to include that data.

Does that help?

REPLY

David Guo January 8, 2020 at 11:55 pm #

Hi Jason,

Thank you for the reply. I had tried to remove t
function became bigger both on train set and t
other univariate forecast method like ARIMA, P
a input for my Xgboost model? Is my thought make sense?

Best regards,
David

REPLY

Jason Brownlee January 9, 2020 at 7:26 am #

Perhaps try it?

REPLY

Guna February 17, 2020 at 6:16 am #

HI Jason,
Hope you are doing good. i have a clarification based on finding lags.I'm trying to convert time series problem into regression. for that how can i find lags based on date,product and location. I mean the order of sorting before finding the lag.
Thanks!

REPLY

Jason Brownlee February 17, 2020 at 7:54 am #

Thanks.

See this:
https://machinelearningmastery.com/time-series-forecasting-supervised-learning/

Manjunath February 25, 2020 at 6:54 am #

REPLY

Hi how to forecast time series data using sliding window method
I have not found any forecast method in this tutorial using sliding windows
Please share link which you have forecast time series data using sliding window method

Jason Brownlee February 25, 2020 at 7:53 a

See the examples here:
https://machinelearningmastery.com/start-here/#de

VU March 1, 2020 at 4:30 pm #

Excellent !

Jason Brownlee March 2, 2020 at 6:15 am #

REPLY

I'm happy to hear that it was helpful!

Leave a Reply

Name (required)

Email (will not be published) (required)

Website

SUBMIT COMMENT

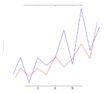
Welcome!

My name is *Jason Brownlee* PhD, and I **help developers** get results with **machine learning**.

Read more

Never miss a tutorial:

Picked for you:



How to Create an ARIMA Model for Time Series Fo

How to Convert a Time Series to a Supervised Learning Problem in Python

Time Series Forecasting as Supervised Learning

11 Classical Time Series Forecasting Methods in Python (Cheat Sheet)

How To Backtest Machine Learning Models for Time Series Forecasting

Loving the Tutorials?

The Time Series with Python EBook
is where I keep the *Really Good* stuff.

SEE WHAT'S INSIDE